2015/2016

Carte Microcontrôleur

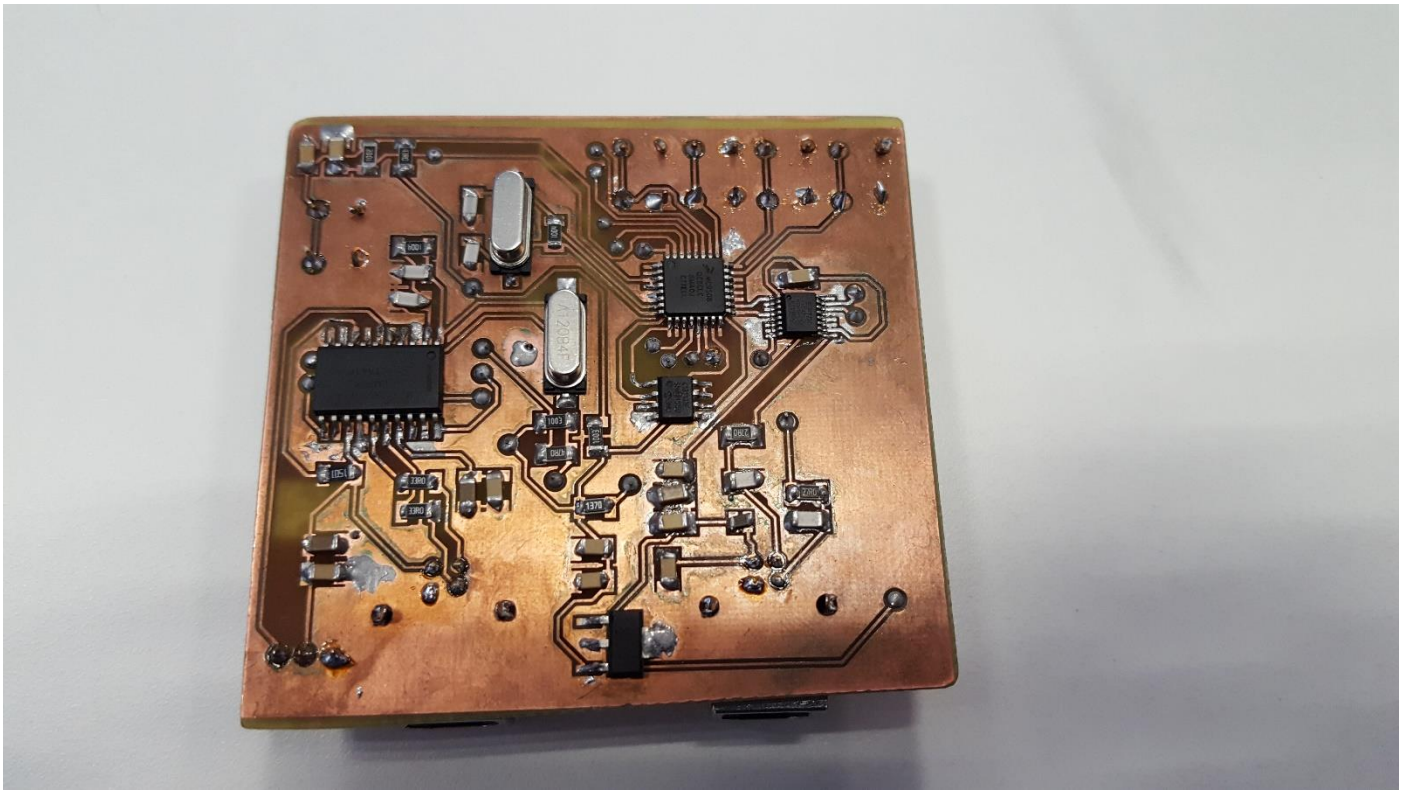
Compte Rendu

Clément Calliau - Pierre Louis Renaud
PREPA SER 1
ESTEI

Table des matières

Présentation du projet

Pour clôturer notre année en conception électronique, nous devions concevoir une carte microcontrôleur, qui propose une partie des fonctionnalités de la carte de développement Freescale utilisée en architecture système. Dans ce projet, une nouvelle technologie de fabrication nous a été proposée : le CMS (composant monté en surface). Celle-ci permet d'augmenter la densité de composants, donc de réduire la carte, mais augmente la difficulté de production. Cette carte est équipée de plusieurs sections, certaines permettant son bon fonctionnement ainsi que d'autres, servant à communiquer.

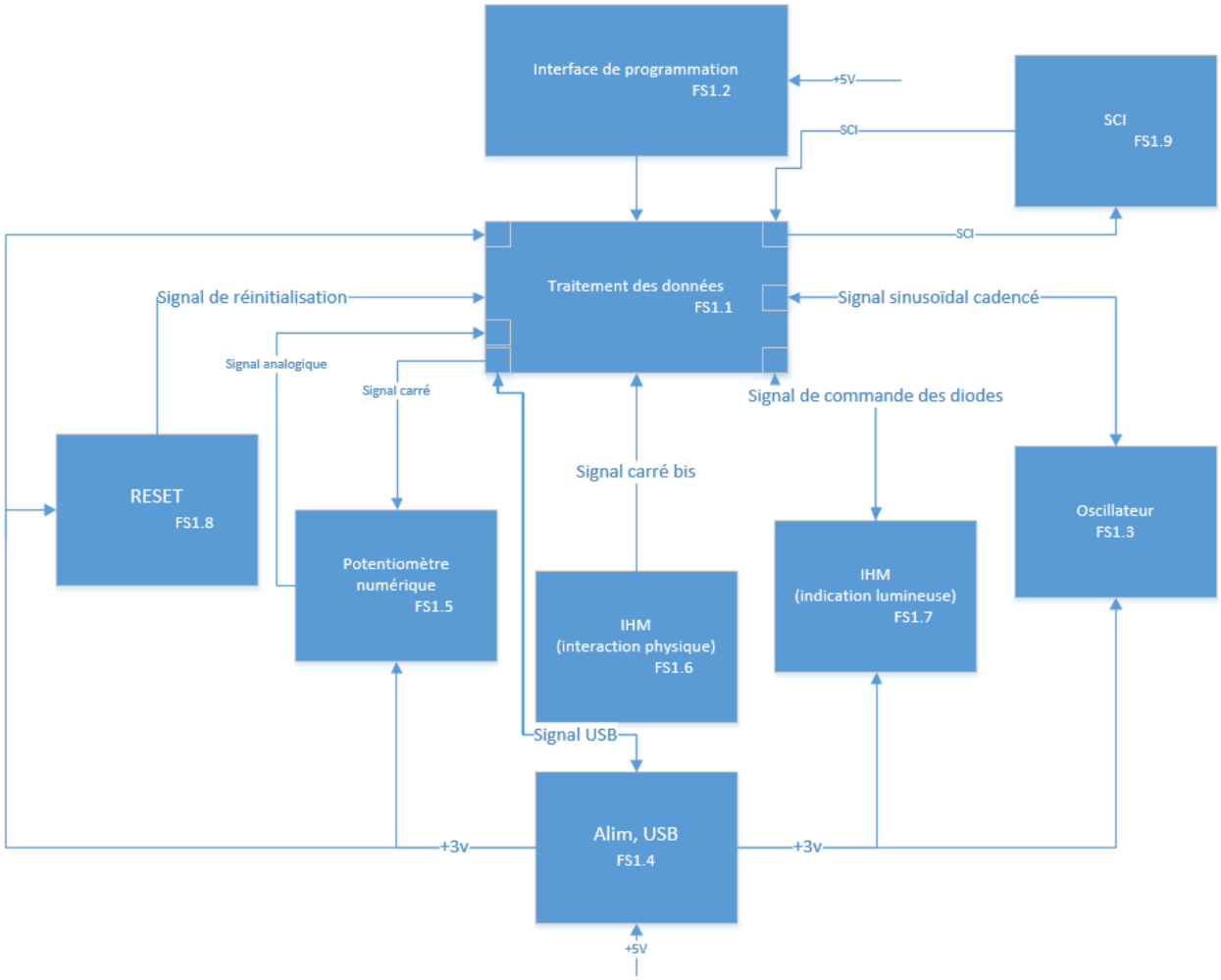


Analyse fonctionnelle

1. *Schéma fonctionnel niveau 1*

FP1 : Exécution d'un programme.

2. *Schéma fonctionnel niveau 2*



Fonctions niveau 2

FS1.1 : Exécution d'un programme.

FS1.2 : Flash d'un programme dans le microcontrôleur.

FS1.3 : Générateur d'horloge externe du microcontrôleur.

FS1.4 : Permet d'alimenter et de communiquer avec le pc et la carte microcontrôleur via USB.

FS1.5 : Transforme une infirmation SPI en valeur analogique.

FS1.6 : Interface homme machine, envoi d'informations au microcontrôleur.

FS1.7 : Interface homme machine, envoi d'informations depuis le microcontrôleur.

FS1.8 : Remet le microcontrôleur à son état par défaut.

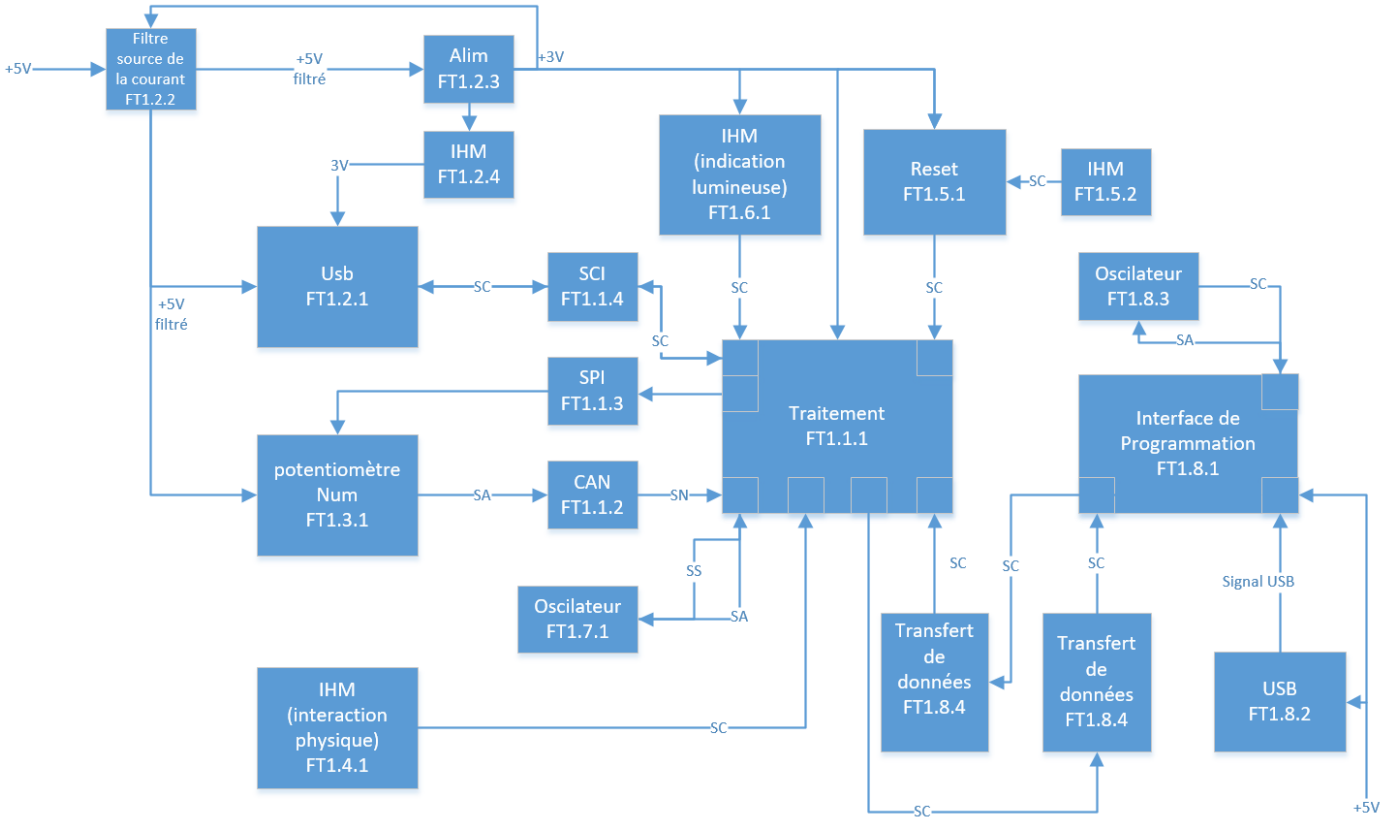
FS1.9 : Permet de transmettre des données entre deux appareils.

FS1.10 : Envoi du programme à l'interface de programmation.

Description des signaux

Signaux	**Forme**	**Commentaire**
+3V	Continue	Tension d'alimentation filtrée
+5V	Continue	Tension d'alimentation USB
Signal sinusoïdal cadencé.	Sinusoïdal	Signal de l'horloge du microcontrôleur.
Signal de réinitialisation.	Continue	Signal demandant au microcontrôleur de se remettre à état par défaut.
Signal USB.	Carré	Signal de communication et de transfert de donné entre un pc et le microcontrôleur.
Signal carré bis.	Carré	Signal émis par les BP.
Signal carré.	Carré	Signal de commande.
Signal analogique		Signal émis par le potentiomètre
Signal de commande des diodes	Carré	Signaux permettant d'allumer ou éteindre les diodes
SCI	Carré	Signal de transmission
Programme	Carré	Code objet du programme

3. Schéma fonctionnel niveau 3



Fonction niveau 3

FT1.1.1 traite les données.

FT1.1.2 utilisation de la fonction CAN

FT1.2.1 utilisation des bus de communication.

FT1.2.2 suppression du bruit.

FT1.2.3 transforme la tension 5V en 3 V.

FT1.2.4 IHM, indications visuelles.

FT1.3.1 permet l'utilisation et le réglage du potentiomètre numérique.

FT1.4.1 IHM.

FT1.5.1 reset de la carte lors de la mise sous tension.

FT1.1.3 utilisation de la fonction SPI

FT1.1.4 utilisation de la fonction SCI

FT1.5.2 Reset de la carte par action physique

FT1.6.1 IHM, indications visuelles.

FT1.7.1 cadence le microcontrôleur.

FT1.8.1 permet de programmer le microcontrôleur.

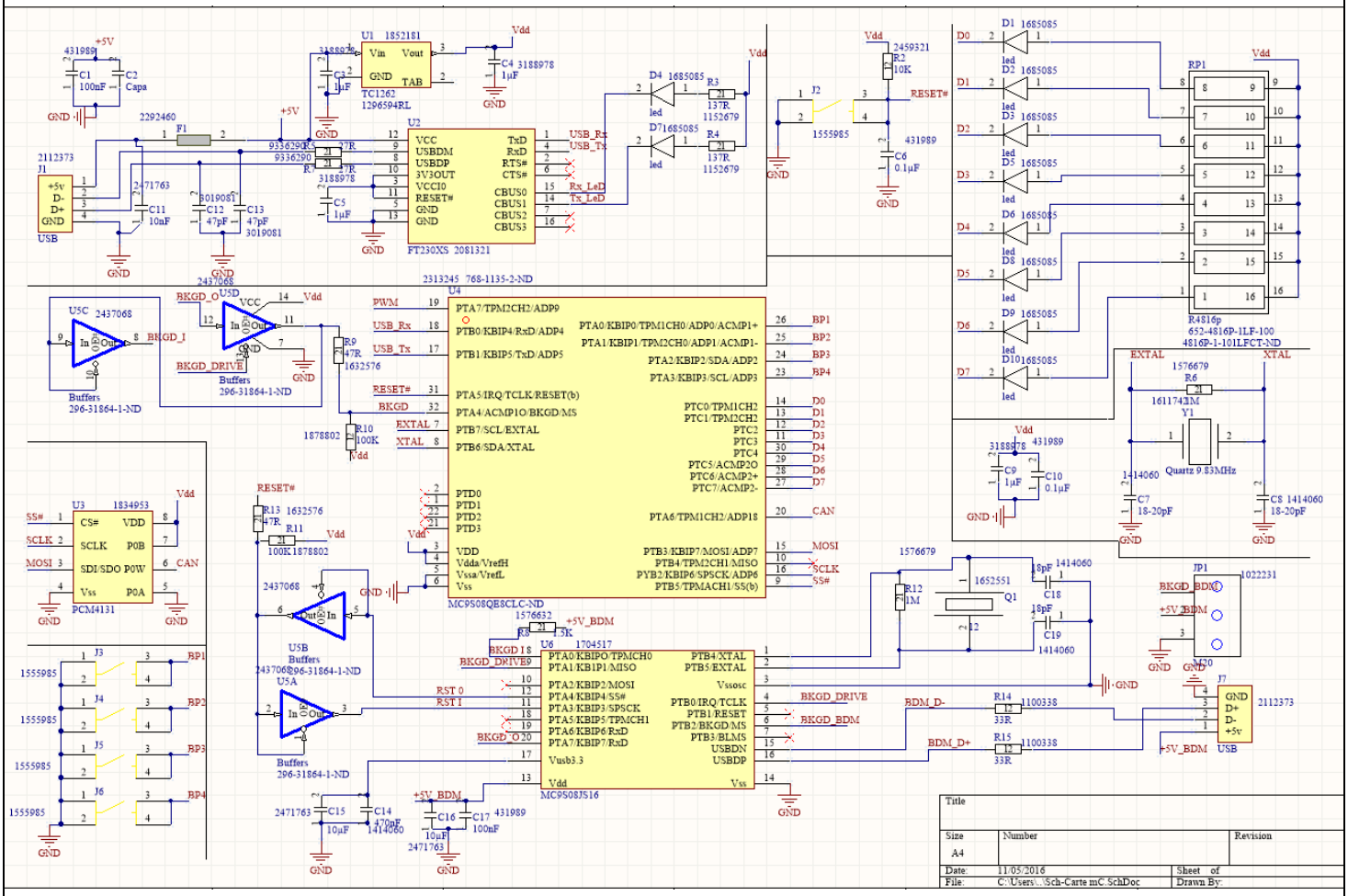
FT1.8.2 alimente et transfert les données au programmateur.

FT1.8.3 cadence le programmateur.

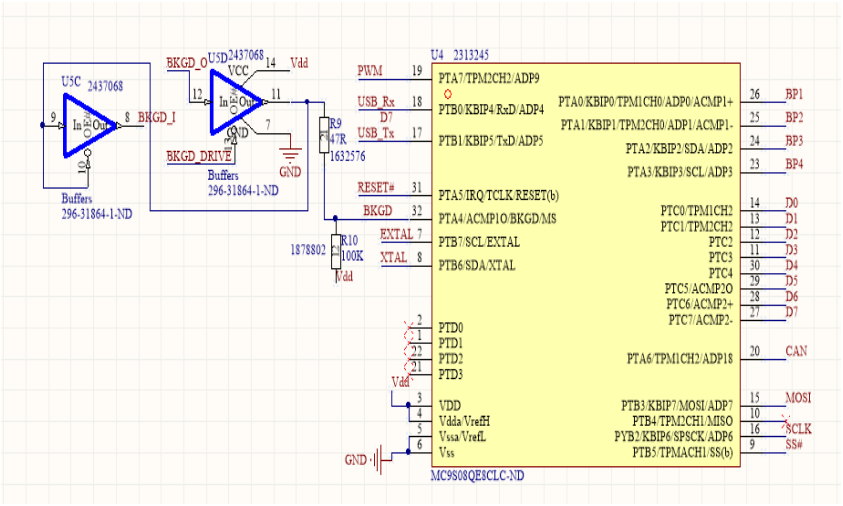
FT1.8.4 transfert les données du programmateur au microcontrôleur.

Analyse structurelle

1. Schéma structurel

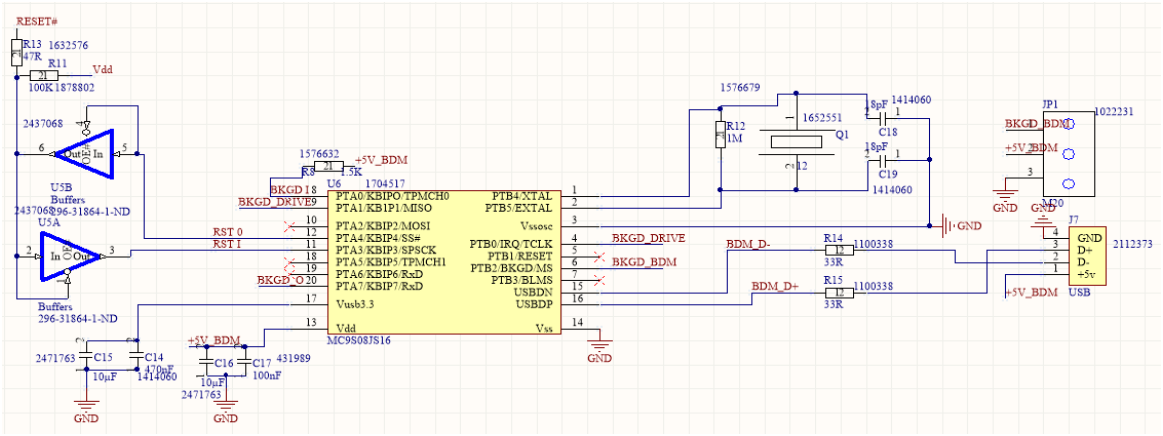


2. *Analyse structurelle*



Microcontrôleur

Ce microcontrôleur est le cœur de la carte. Il est accompagné de deux buffers qui permettent de conserver un état selon la commande BKGD_DRIVE. Ce microcontrôleur est branché aux différentes fonctions offertes par la carte.
Le microcontrôleur utilisé, le MC9S08QE8CLC de NXP, possède 512 octets de ram, 8Ko de mémoire flash et 32 broches et fonctionne sous l'architecture VON NEUMANN.



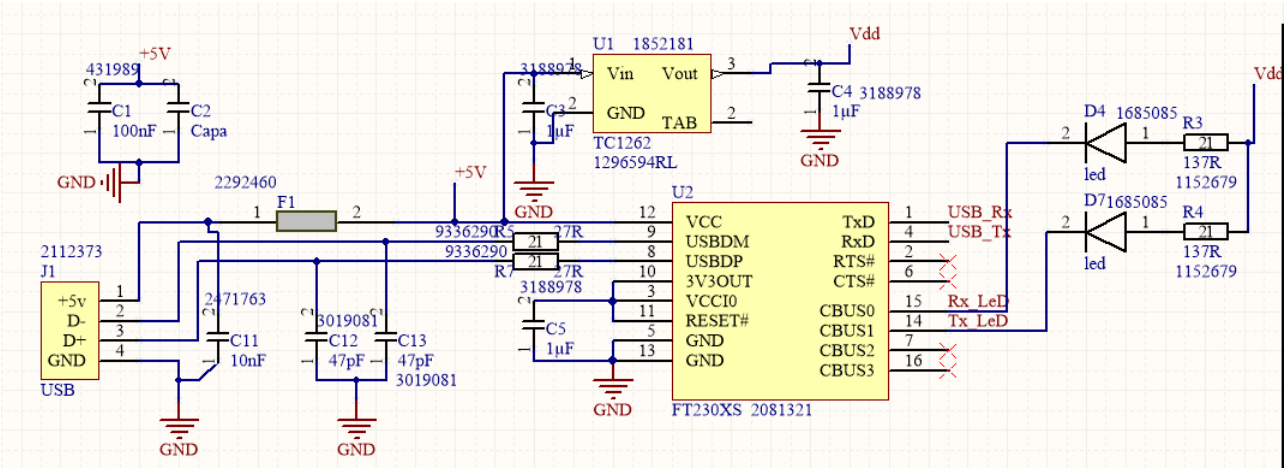
Programmateur

Ce circuit est le programmateur intégré de la carte microcontrôleur, il permet d'envoyer le programme au microcontrôleur. Il possède sa propre horloge de 12Mhz. Les sorties du montage peuvent garder un état, en entrée ou en sortie, grâce aux buffers U5A et U5B.

On a également 4 condensateurs de découplage C15, C14/C16, C17 :
On peut calculer leurs fréquences de coupure avec la formule :

$$\frac{1}{2\pi RC}$$

Au niveau de l'horloge du circuit, on retrouve deux condensateurs de découplage C18, C19 connectés entre le quartz ainsi qu'une résistance R12.



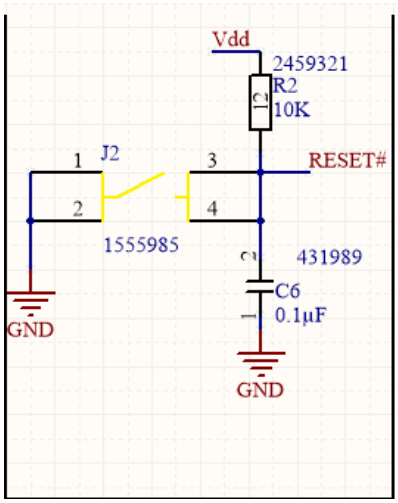
Alim, USB,SCI

Ce circuit sert d'alimentation et d'interface de transfert entre le microcontrôleur et l'ordinateur.
On peut également utiliser la fonction SCI entre le microcontrôleur et un périphérique USB.
On retrouve également des condensateurs de découplage C11, C12, C13, C1, C2, C3, C5.
On calcule leurs fréquences de coupure avec la formule : $\frac{1}{2\pi RC}$.
On détaillera le calcul du condensateur de découplage C5 :
$\frac{1}{2\pi RC}=\frac{1}{2\pi 0.5*1\mu}$=318Mhz
Avec :
R la résistance du fil mesuré sur la carte (0.5 ohm)
C le condensateur C5 (1µF)

Le composant TC1262 est un régulateur de tension. Il prend en entrée 5V et donne en sortie 3V
On peut calculer sa puissance dissipée par :

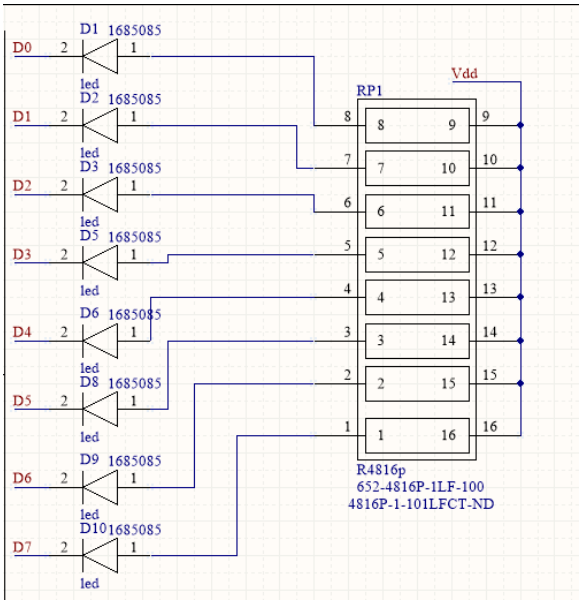
Pd = (Vin_{max}-$Vout_{min}$)*Iload max
Pd = (5-3±2.5%)*0.500
Pd =2.45W

Le composant FT230XS :
Il gère les connexions entre le PC et la carte microcontrôleur. Il contient le micro-logiciel pour être reconnu par le PC au branchement USB.



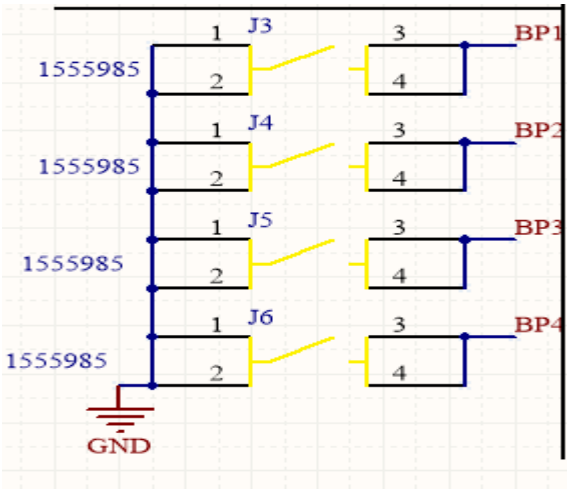
Fonction Reset

La broche ‘reset’ a pour fonction la réinitialisation du microcontrôleur dans son état par défaut. Son activation doit respecter une période minimun à l’état bas (logique inverse) de 100ns.



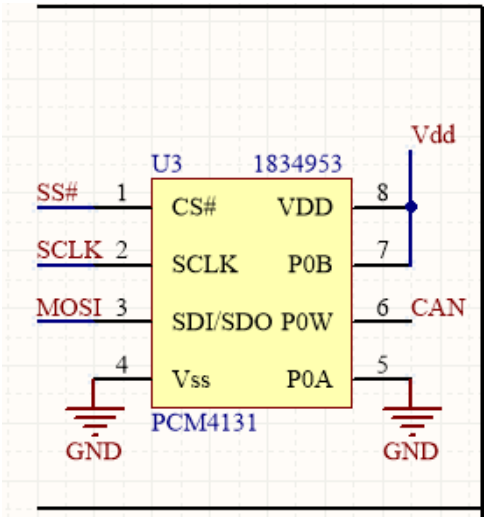
Fonction IHM (led)

La datasheet indique que la LED absorbe au max 10mA et fonctionne a 1.95V
On a également un VDD à 3V.
<u>On applique la maille :</u>
VDD-VR-Vled=0
Vdd-(I*R)-Vled=0
$\frac{3-1.95}{10mA}$=135ohm
L’utilisation d’une résistance normalisée de 100Ohm est cohérente.



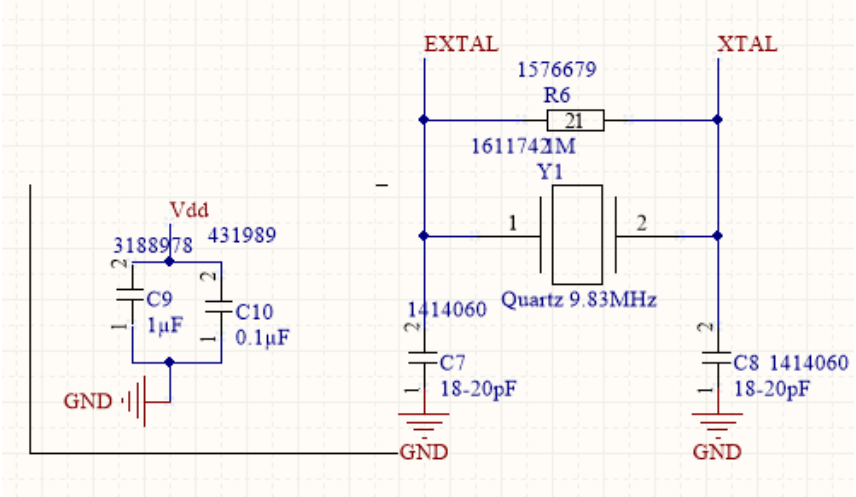
Fonction IHM (bouton poussoir)

Ce montage permet d'envoyer des informations logiques aux broches du microcontrôleur.



Fonction potentiometre

Ce potentiomètre numérique est réglable via la sortie SPI du microcontrôleur sur la broche MOSI. Il renvoie une valeur analogique à la broche CAN du microcontrôleur.



Fonction Oscillateur

Ce circuit sert d'horloge pour le microcontrôleur. On cadence le microcontrôleur à 9.83 MHz
On retrouve également deux condensateurs de découplage.
On calcule leurs fréquence de coupure avec la formule :

$$\frac{1}{2\pi RC}$$

C avec C7/C8 à 18pF.
Et R avec R6 à 1Mohm.

Egalement deux condensateur de decouplage C9 et C10.

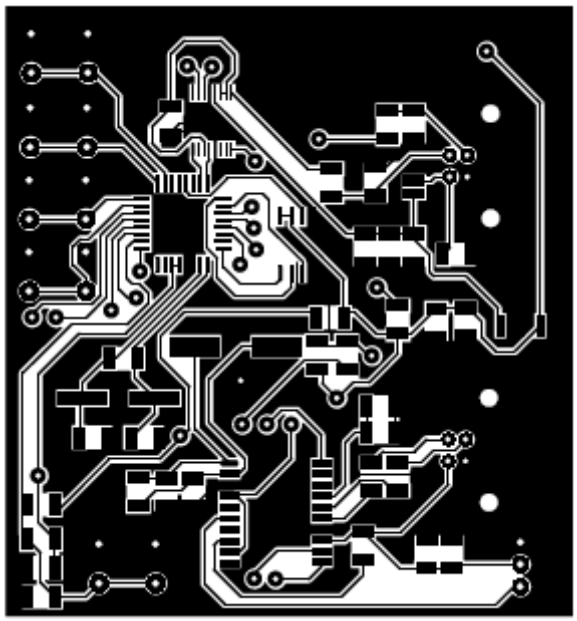
Fabrication

1. Nomenclature

Designateur	Description	Caractéristique	Température de fonctionnement	Fournisseur	Référence	Quantités	Prix UHT	Prix TTL HT
C1, C17	Condensateurs	50V max/100nF	-30°C à +85°C	Farnell	CC0603MRY5V9BB104	2	0,0169	0,0338
C2	Condensateurs	50V max/4,7µF	-55°C à +85°C	Farnell	C1206C475K5PACTU	1	0,778	0,778
C3, C4, C5, C9	Condensateurs	16V max/1µF	-25°C à +85°C	Farnell	CC1206MRY5V7BB105	4	0,123	0,492
C6, C10	Condensateurs	50V max/0.1µF	-30°C à +85°C	Farnell	CC0603MRY5V9BB104	2	0,169	0,338
C7, C8	Condensateurs	50V max/ 18-20pF	-55°C à +125°C	Farnell	C0603C200J5GACTU	2	0,209	0,418
C11	Condensateurs	250V max/10nF	-55°C à +125°C	Farnell	C1206X103JAGACAUTO	1	0,749	0,749
C12, C13	Condensateurs	50Vmax /47pF	-55°C à +125°C	Farnell	CC0402CRNPO9BNR47	2	0,0403	0,0806
C14	Condensateurs	50Vmax /470nF	-55°C à +125°C	Farnell	C1206X474K5RACTU	1	0,989	0,989
C15, C16	Condensateurs	250V max/10µF	-55°C à +125°C	Farnell	C1206X103JAGACAUTO	2	0,749	1,498
C18, C19	Condensateurs	250Vmax /18µF	-55°C à +125°C	Farnell	0805N180J500CT'	2	0,989	1,978
D1, D2, D3, D4, D5, D6, D7, D8, D9, D10	LED	10mA max	-40°C à +85°C	Farnell	SML-811WTT86AAP	10	0,138	1,38
F1	Ferrite	impedance 600ohm/ 1,5A max	250°C' max	Farnell	MI0805K601R-10	1	0,128	0,128
J1, J7	Connecteur USB-B	USB 2,0'	/	Farnell	USB-B-S-RA	2	0,35	0,7
J2, J3, J4, J5, J6	Boutton poussoir	50mA max /24Vmax	/	Farnell	FSM8JH	5	0,117	0,585
JP1	GPIO	3A max	/'	Farnell	M20-9980346	1	0,158	0,158
Q1	Quartz	12MHz/100Vmax'	-20°C à +70°C	Farnell	ABLS-12.000MHZ-B2-T	1	0,363	0,363
R2	Resistances N-1206	10k ohm/	-55°C à +150°C	Farnell	RC1206FR-0710KL	1	0,0781	0,0781
R3, R4	Resistances N-1206	137ohm	-55°C à +150°C	Farnell	CRCW1206137RFKEA	2	0,0857	0,1714
R5, R7	Resistances N-1206	200V max / 27 ohm/	-55°C à +155°C	Farnell	MC0125W1206127R	2	0,0034	0,0068
R6, R12	Resistances N-1206	200V max / 1Mohm	-55°C à +155°C	Farnell	MCHP06W2F1004T5E	2	0,212	0,424
R8	Resistances N-1206	200Vmax / 1,5Kohm	-55°C à +155°C	Farnell	MCHP06W2F1501T5E	1	0,208	0,208
R9, R13	Resistances N-1206	47ohm /200Vmax	-55°C à +155°C	Farnell	MC1206S4F470JT5E	2	0,072	0,144
R10, R11	Resistances N-1206	100k ohm /200Vmax	-55°C à +155°C	Farnell	CRCW1206100KFKEB	2	0,0857	0,1714
R14, R15	Resistances N-1206	200Vmax/ 0,33ohm	-55°C à +155°C	Farnell	LR1206-R33FW	2	0,441	0,882
RP1	Réseau de résistances	50Vmax/100ohm	-55°C à +125°C	Mouser	4816P-1-101LF	1	1,1	1,1
U1	Régulateur de tension	6Vmax/	-40°C à +125°C	Farnell	TC1262-3.0VDB	1	0,528	0,528
U2	Contrôleur USB	5,5Vmax /100mA max	-40°C à +85°C	Digi-Key	FT230XS-R	1	2,12	2,12
U3	Potentiometre numerique	5,5Vmax /10Kohm ± 20%	-40°C à +125°C	Farnell	MCP4131-103E/SN	1	0,695	0,695
U4	Micro-controleur	3,6V max	-40°C à +85°C	Farnell	MC9S08QE8CLC-ND'	1	2,33	2,33
U5	CI 4 buffers	5,5Vmax	-40°C à +85°C	Digi-Key	SN74LV125ADR	1	0,41	0,41
U6	MC9S08JS16	5,5Vmax	-55°C à +150°C	Farnell	MC9S08JS16CWJ	1	2,4	2,4
Y1	Quartz	9,83MHz/100Vmax'	-10°C à 60 °C	Farnell	HCM49 9.8304MABJ-UT	1	0,719	0,719
							Total :	23,0561

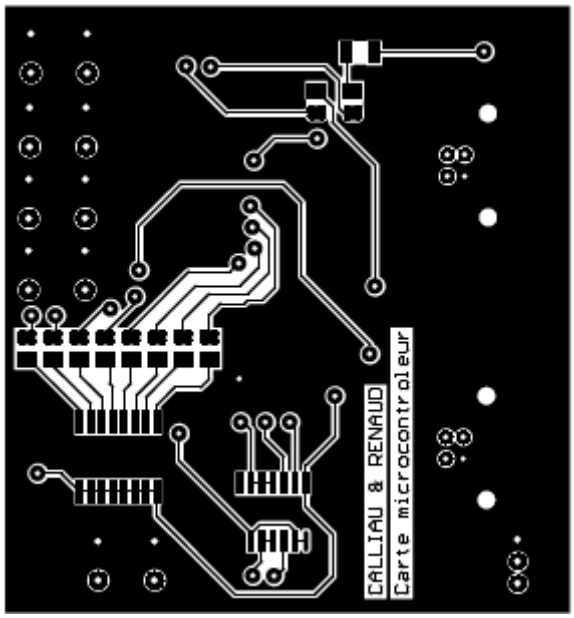
2. Plan

Top :



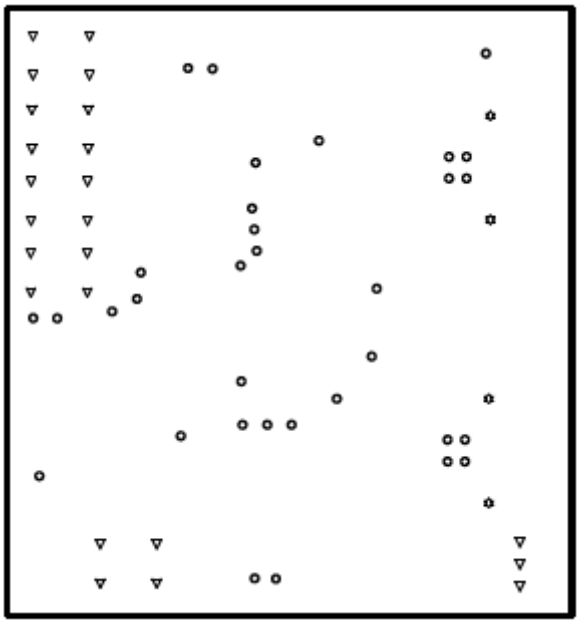
Echelle 1:1

Bottom :



Echelle 1:1

Plan de perçage :

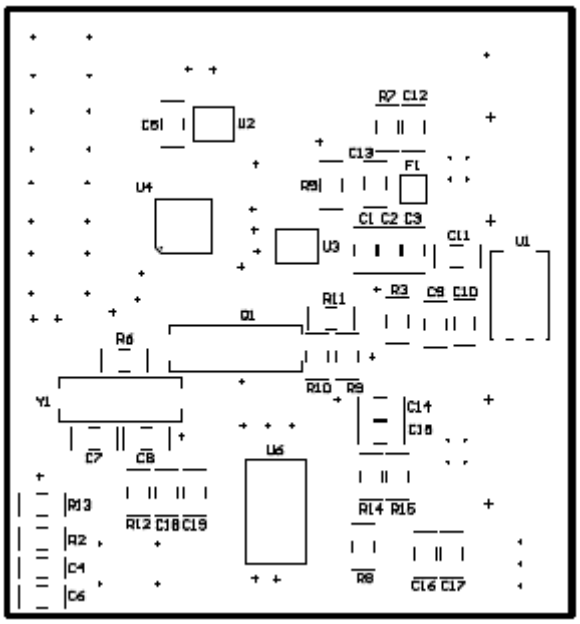


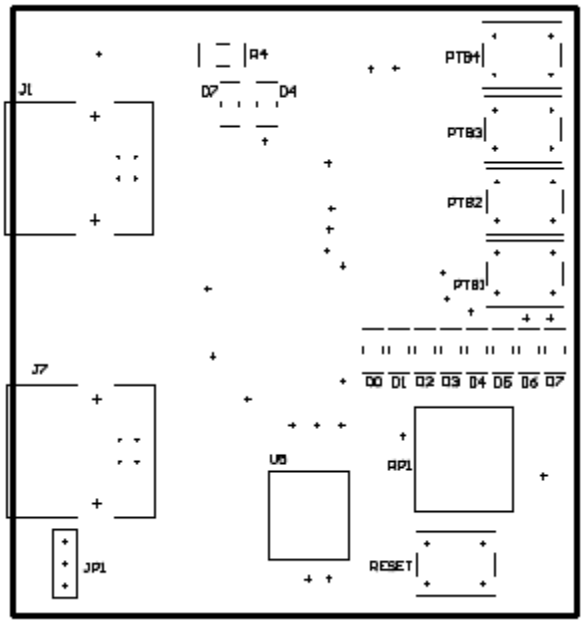
Echelle 1:1

Ronds : 0.6

Croix : 2 mm
Triangles : 0.8 mm

Implantation des composants :





Echelle 1:1

3. *Test de fonctionnement*

Pour s'assurer du bon fonctionnement de la carte, plusieurs tests doivent être faits au cours de sa production pour éviter des erreurs difficilement corrigeables.

La première vérification se fait sur les empreintes des bibliothèques PCB. Une feuille papier est imprimée avec les empreintes de tous les composants et on vérifie la correspondance de taille et de positionnement des broches.

Après le tirage de la carte, le premier test est visuel. Il s'agit de s'assurer qu'aucun faux contact ou court-circuit visible à l'œil nu n'est présent.

Une fois les défauts visibles corrigés, nous utilisons un multimètre pour des tests ohmiques. Deux conducteurs sont touchés par les pointes de test et l'outil buzz si un contact existe entre les deux conducteurs. Ce test permet de repérer à la fois les court-circuit et les microcoupures invisibles à l'œil nu. Par la suite, un test ohmique a été effectué sur chaque patte et la masse afin de vérifier que l'on ait une grande valeur ohmique.

Test de validation de la carte

Pour tester le bon fonctionnement général de la carte microcontrôleur, nous avons réalisé deux programmes qui englobent la totalité des fonctionnalités hardware présentes.

Le premier programme vérifie le fonctionnement des boutons poussoirs, des LEDS et de la communication SCI.

Le second programme vérifie la conversion analogique-numérique, la communication SPI et les LEDS.

Etudes de registre :

Registres de configuration de la fonction SCI :

SCIC1 : = $00

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

LOOPS[7] 0 : Mode double broche
SCISWAI[6] 0 : Pas de gestion de la veille
RSRC[5] 0 : Sans effet
M[4] 0 : Envoi de 8 bits de data
WAKE[3] 0 : Pas utilisé
ILT[2] 0 : Pas utilisé
PE[1] 0 : Pas de test de parité
PT[0] 0 : pas utilisé

SCIC2 : = $00

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

TIE[7] 0 : Pas d'interruptions
TCIE[6] 0 : Pas d'interruptions
RIE[5] 0 : pas d'interruptions
ILIE[4] 0 : pas d'interruptions
TE[3] 0 : Activation de la transmission
Doit être mis à 1 pour commencer l'envoi
RE[2] 0 : non-activation de la réception
RWU[1] 0 : Pas de réveil à la réception.
SBK[0] 0 : Transmission normale

SCIS1 : full flag

7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x

TDRE[7] : Flag non utilisé
TC[6] : Flag à 1 en fin de transmission.
RaZ : écrire dans SCID
RDRF[5] : Flag de fin de réception. RaZ lire RDRF et SCID
IDLE[4] : Pas de idle
OR[3] : Flag de réception écrasée
NF[2] : Pas de détection bruit
FE[1] : Pas de détection d'erreur
PF[0] : Pas de détection de parité

SCIBD : 0000 0000:0001 1011

12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	1	0	1	1

SBR [12:0] 0000-0001 1011 : 9700baud

Tous les autres bits sont laissés à leur état par défaut.

Registres de configuration de la fonction CAN :

ADCSC1 : $21

7	6	5	4	3	2	1	0
0	0	1	0	0	0	0	1

COCO[7] : Flag à 1 quand une conversion est complète.
RAZ : lire ADCRL
AIEN[6] : 0 : La fin ne provoque pas d'interruption
ADC0[5] : 0 : conversions pas en continu
ADCH[4:0] : 01000 : CAN branché sur PTA6
Ecrire dans ce registre active automatiquement une conversion

ADCSC2 : $00

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

ADACT[7] : Flag à 1 quand une conversion est en cours
ADTRG[6] : 0 : Conversion lancée par le programme
ACFE[5] : 0 : pas de comparaison
ACFGT[4] : 0 : pas de comparaison

ADCCFG : 0110-0000 = $60

7	6	5	4	3	2	1	0
0	11		0	00		00	

ADLPC[7] : 0 : Pas de basse consommation
ADIV[6:5] : 11 : Busclock/8
ADLSMP[4] : 0 : Échantillon court
MODE[3:2] : 00 : Conversion sur 8 bits
ADICLK[1:0] : 00 : Busclock

ADCRH : DATA 4 bits hauts
ADCRL : DATA 8 bits bas

15	14	13	12	11	10	9	8
x	x	x	x	0	0	0	0

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

ADR[11:8] : Registre 8b, contient la partie haute du résultat de la CAN

ADR[7:0] : Registre 8b, contient la partie basse du résultat de la CAN
Utilisation de la partie basse uniquement, 8 bits.

Tous les autres bits sont laissés à leur état par défaut.

Registres de configuration de la fonction SCI :

SPIC1 : $12

7	6	5	4	3	2	1	0
0	0	0	1	0	0	1	0

SPIE[7] 0 : Pas de réception
SPE[6] 0 : Garde l'envoi actif en cas d'interruption
SPTIE[5] 0 : Pas d'interruption
MSTR[4] 1 : Micro en maître
CPOL[3] 0 : Actif haut
CPHA[2] 0 : Premier front d'horloge au milieu du premier bit
SSOE[1] 1 : Sélection d'esclave auto
LSBFE[0] 0 : MSB first

SPIC2 : $10

7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	0

MODFEN[4] 1 : Gestion de la broche de sélection d'esclave par la fonction SPI
BIDIROE[3] 0 : I/O par des broches séparées
SPISWAI[1] 0 : Pas d'arrêt en mode pause
SPC0[0] 0 : Non utilisé

SPIS : Full flag

7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x

SPRF[7] : Flag de réception complète, non utilisé
SPTEF[5] : Flag à 1 quand le registre d'envoi est vide. RaZ en lisant SPTEF + écrire dans SPID
MODF[4] : Pas de gestion des erreurs

SPIBR : $12

7	6	5	4	3	2	1	0
0	0	0	1	0	0	1	0

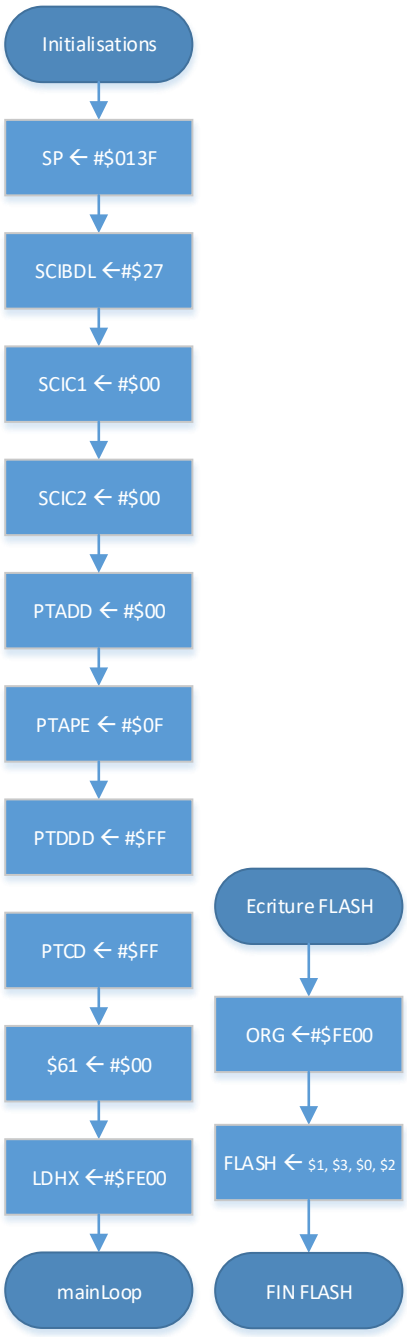
SPPR[6:4] 001 : Pré-division du busclock
SPR[3:0] 0010 : Seconde division du busclock
Baudrate/16

Tous les autres bits sont laissés à leur état par défaut.

1. *Premier programme : BP-LED-SCI*

Ce programme attend une combinaison prédéfinie de pressions successives sur les boutons poussoirs pour transmettre une valeur par bus SCI. En cas de mauvaise combinaison, une alerte lumineuse est renvoyée.

Algorigramme



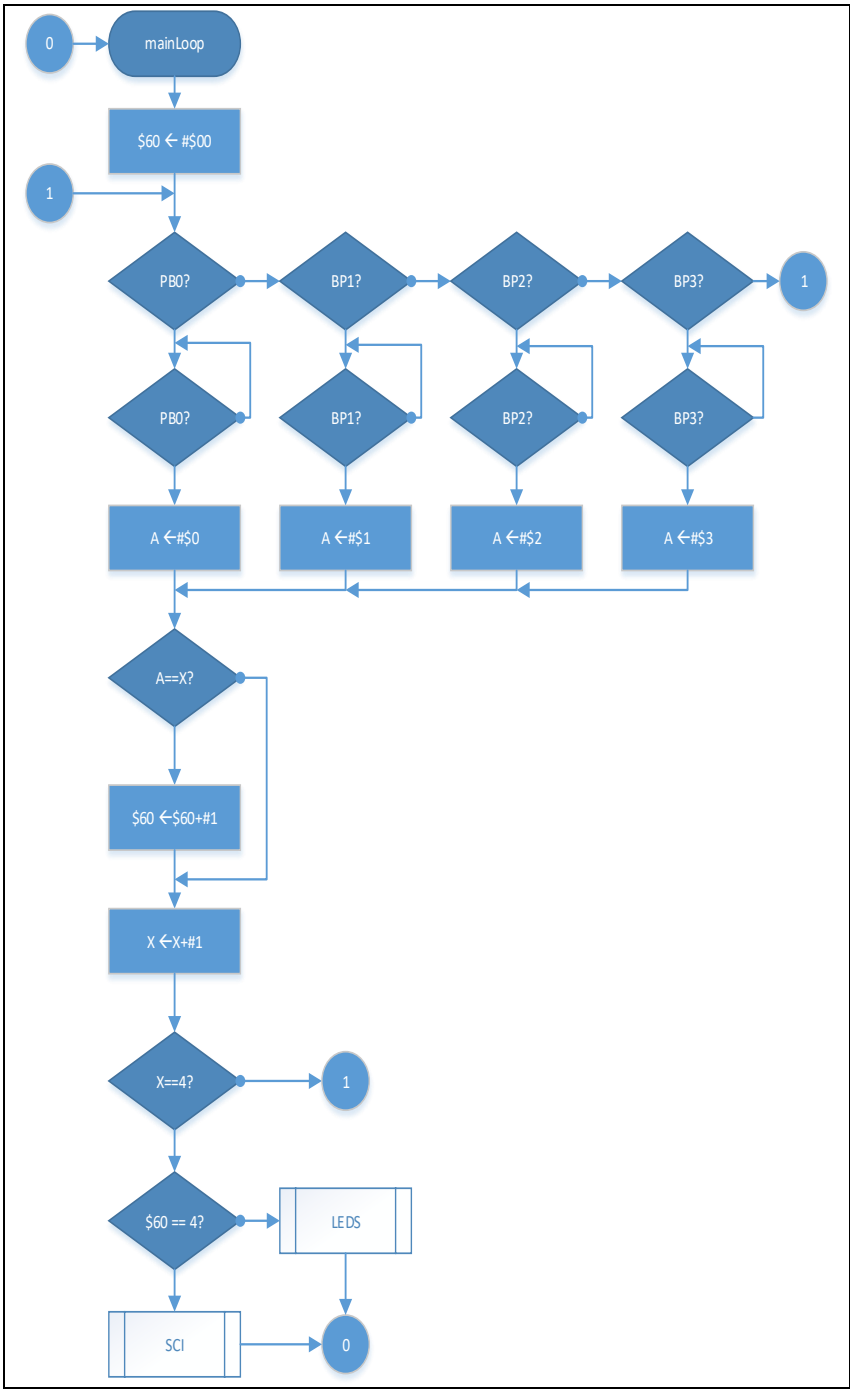
Le programme est dépendant de plusieurs initialisations.

La première se passe lors de l'envoi du programme dans le microcontrôleur : on écrit dans la mémoire flash la combinaison attendue par la pression des boutons poussoirs.
Cette partie du programme est à la toute fin du code source pour éviter les erreurs d'effacement de code déjà écrit.

La seconde initialisation gère les espaces mémoires et registres utilisés par le programme :

- Sommet de la pile
- SCI pour l'envoi de valeurs
- Port A pour recevoir les commandes des boutons poussoirs
- Port C pour envoyer les commandes aux LEDS
- Espace mémoire
- Index de la boucle principale.

Cette série d'initialisations est faite dans la première partie du code source, avant le début de la boucle principale.

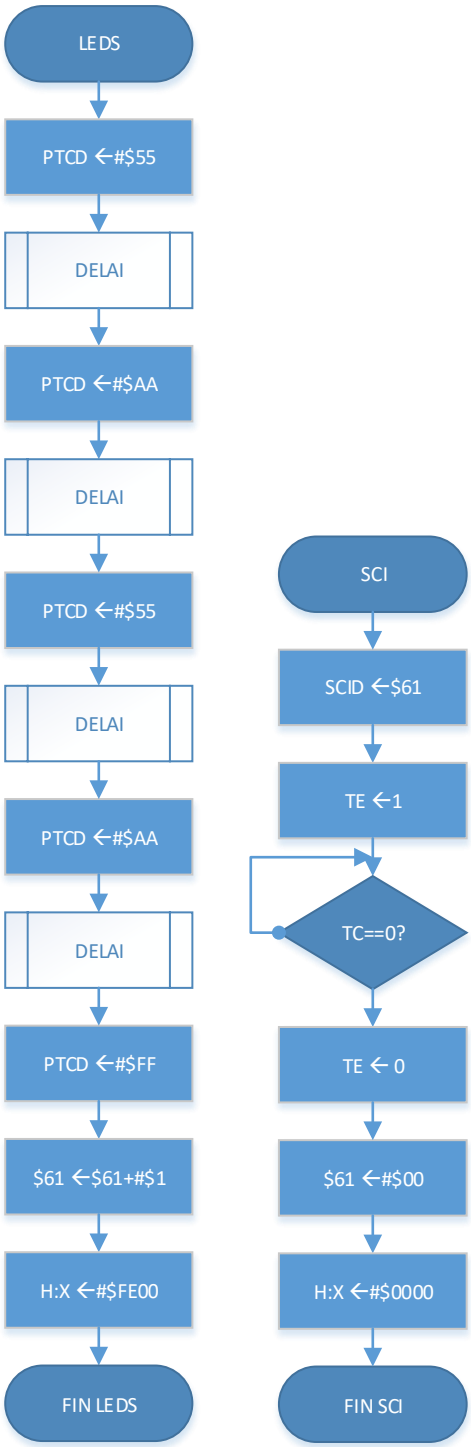


La mainLoop contient la boucle principale du programme, et est divisée en deux parties. Elle gère l'entrée d'une combinaison de pression sur les boutons poussoirs, notés BP, pour débloquer l'accès au reste du programme.

La première partie gère l'attente d'un évènement qui est la pression bouton poussoir, puis il indique dans le registre A quel bouton a été pressé.

La seconde partie compare la valeur dans A, relative au bouton pressé, à la valeur attendue qui est, elle, stockée dans la mémoire flash.

Fonctionnement :
L'attente et la comparaison avec la valeur attendue tournent en boucle en incrémentant l'index à chaque pression, et une mémoire à chaque fois que le BP correspond à la valeur attendue.
Une fois que 4 BP ont été successivement pressés, le programme branche vers le sous-programme SCI si les 4 BP pressés étaient bons, et au sous-programme LEDS si une ou plusieurs erreurs ont été commises. Au retour de ces sous-programmes, la boucle principale se relance.

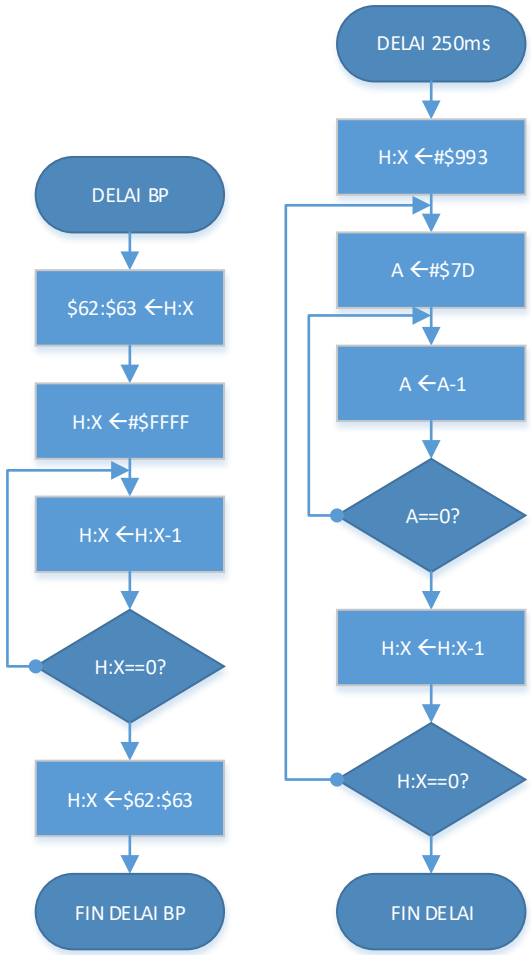


Ces sous-fonctions ne se déclenchent que lorsque que quatre boutons poussoirs ont été pressés successivement.

Si la combinaison entrée sur les boutons n'est pas celle inscrite dans la mémoire flash, le sous-programme déclenche un clignotement des LEDS pour signaler l'erreur.
Le sous-programme appelle un délai après chaque modification des LEDS pour que leur visualisation soit facile.
Son autre utilité est de donner un temps fixe entre deux tentatives de combinaison, afin de ne pas en laisser une à moitié entrée. Ainsi l'utilisateur sait exactement quand une nouvelle combinaison peut être testée. En outre le sous-programme tient un compte des codes erronés.

Si la bonne combinaison est entrée, le sous-programme d'envoi SCI est appelé. Il permet d'envoyer à un périphérique le nombre d'essais ratés pour accéder à cette fonction.

La fin de ces deux sous-programmes réinitialise l'index servant à l'entrée des BP, puis renvoie vers le début de la boucle principale.



Le programme contient deux sous-programmes de délai.

Le premier sert à supprimer les erreurs dues aux fluctuations électriques lors de la pression des BP, c'est une protection anti-rebond, fixe, supplémentaire.

$$16 + (65\,535 * 8) = 524296\ cycles = 10\mu s$$

Le second délai est appelé par le sous-programme LEDS à chaque changement lumineux. Sa durée sert à bénéficier d'environ une seconde, visible, entre chaque tentative de combinaison sur les BP.

$$8 + 2451(4 * 127 + 10) = busclock * 0.25s = 250ms$$

Code source

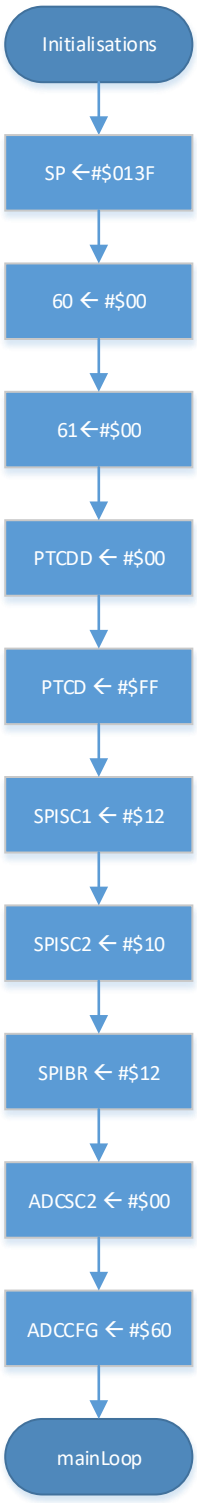
```
; variable/data section
MY_ZEROPAGE: SECTION  SHORT         ; Insert here your data definition

; code section
MyCode:     SECTION
main:
_Startup:
            LDHX   #__SEG_END_SSTACK ; initialize the stack pointer
            TXS
            ; Call generated device initialization function
            JSR    MCU_init
            MOV     #$0027, SCIBDL      ;Paramétrage SCI, 9709 baud.
            MOV     #$00,   SCIC1       ;SCI : Envoi sur 8bits, pas de etst de parité
            MOV     #$00,   SCIC2       ;SCI : Désactivation des interuptions
            MOV     #$00,   PTADD       ;Port A, BP, en entrée.
            MOV     #$0F,   PTAPE       ;Pull-up internes activées sur PTA0-1-2-3.
            MOV     #$FF,   PTCDD       ;Port C, LEDS, en sortie
            MOV     #$FF,   PTCD        ;Extinxion des LEDS
            MOV     #$00,   $61         ;Init de la mémoire de verif boucle
            LDHX    #$FE00              ;Init du poiteur d'ordre des boutons

mainLoop:
            MOV     #$00,   $60         ;Init de la mémoire de verif code
DTEST:
            BRCLR   0, PTAD, BP0        ;Branche si le BP0 est pressé
            BRCLR   1, PTAD, BP1        ;Branche si le BP1 est pressé
            BRCLR   2, PTAD, BP2        ;Branche si le BP2 est pressé
            BRCLR   3, PTAD, BP3        ;Branche si le BP3 est pressé
            BRA     DTEST               ;Si aucun n'est pressé, recommence les tests
BP0:        JSR     SecBP               ;Court délai pour éviter les fluctuations electriques
            BRCLR   0, PTAD, BP0        ;Attend le relachement du BP0
            LDA     #$0                 ;Charge dans A l'info "BP0 pressé"
            BRA     FTEST               ;Branche vers la fin des tests BP
BP1:        JSR     SecBP               ;Court délai pour éviter les fluctuations electriques
            BRCLR   1, PTAD, BP1        ;Attend le relachement du BP1
            LDA     #$1                 ;Charge dans A l'info "BP1 pressé"
            BRA     FTEST               ;Branche vers la fin des tests BP
BP2:        JSR     SecBP               ;Court délai pour éviter les fluctuations electriques
            BRCLR   2, PTAD, BP2        ;Attend le relachement du BP2
            LDA     #$2                 ;Charge dans A l'info "BP2 pressé"
            BRA     FTEST               ;Branche vers la fin des tests BP
BP3:        JSR     SecBP               ;Court délai pour éviter les fluctuations electriques
            BRCLR   3, PTAD, BP3        ;Attend le relachement du BP3
            LDA     #$3                 ;Charge dans A l'info "BP3 pressé"
FTEST:      CMP     0,X                 ;Compare Le bouton pressé à la valeur attendue
            BNE     NON                 ;Saute si le bouton est mauvais
            INC     $60                 ;Si le bouton est bon, incrémente la vérification
NON:        INCX                        ;Incrémente le compteur de test
            CPX     #$4                 ;Vérifie si le bon nombre de BP à été pressé
            BNE     DTEST               ;Si non, repart tester le BP suivant
            LDA     #$4                 ;Si assez de BP ont étés pressés
            CMP     $60                 ;Vérifie combien de bons boutons ont été pressés
            BNE     LEDS                ;Si moins de 4 bons boutons pressés, branche vers l'alerte visuelle
            BRA     SCI                 ;Si 4 bons boutons pressés, envoie la donnée SCI
```

```
LEDS:       MOV     #$55,   PTCD    ;Clignotement les LEDS. Sous-programme d'avertissement : MDP faux.
            JSR     DELAI               ;Attend 250ms
            MOV     #$AA,   PTCD        ;Eteint les LEDS
            JSR     DELAI               ;Attend 250ms
            MOV     #$55,   PTCD        ;Allume les LEDS
            JSR     DELAI               ;Attend 250ms
            MOV     #$AA,   PTCD        ;Eteint les LEDS
            JSR     DELAI               ;Attend 250ms
            MOV     #$FF,   PTCD        ;Eteint les LEDS
            INC     $61                 ;Incrémente le compte du nombre d'essais ratés
            LDHX    #$FE00              ;Reset le pointeur de BP
            BRA     mainLoop            ;Recommence une nouvelle série de tests


SCI:        MOV     $61,    SCID    ;Transfert le nombre de tests manqués via SCI
            MOV     #$08,   SCIC2       ;Active la transmission
TRANS:      BRCLR   6,  SCIS1,  TRANS   ;Attend la fin de l'envoi
            MOV     #$00,   SCIC2       ;Stoppe la transmission
            MOV     #$00,   $61         ;Reset les essais ratés
            LDX     #$00                ;Reset du prochain test BP
            BRA     mainLoop            ;Retourne aux tests de BP


DELAI:      LDHX    #$993           ;Configure la boucle externe du délai
LOOPX:      LDA     #$7D                ;Configure la boucle interne
LOOPA:      DECA                        ;Décrémente la boucle interne
            BNE     LOOPA               ;Boucle tant que A n'est pas vide
            AIX     #-1                 ;Décrémente la boucle externe
            CPHX    #$0000              ;Test la fin de la boucle externe
            BNE     LOOPX               ;Si la boucle est finie
            RTS                         ;Met fin au sous-programme

SecBP:      STHX    $62             ;Délai pression des BP, Sécurité supplémentaire à l'enti-rebond. Sauvegarde l'index
            LDHX    #$FFFF              ;Initialisation de la valeur du délai
LOOPB:      AIX     #-1                 ;Décrémentation de la valeur
            CPHX    #$0000              ;Test de fin du délai
            BNE     LOOPB               ;Boucle tant que la valeur est non nulle
            LDHX    $62                 ;Restauration de l'index
            RTS                         ;Retour à la gestion du MDP BP


            ORG     $FE00               ;Décale l'origine de la suite du programme vers la fin de la flash
            FCB     $1, $3, $0, $2      ;Ecrit dans la flash la combinaison de BP demandée pour activer la SCI

```

2. *Programme 2 : CAN-SPI-LED*

Ce programme envoie une valeur par bus SPI à un potentiomètre numérique qui renvoie une valeur analogique. Cette valeur est traitée pour en afficher une indication visuelle sur les LEDS de la carte.

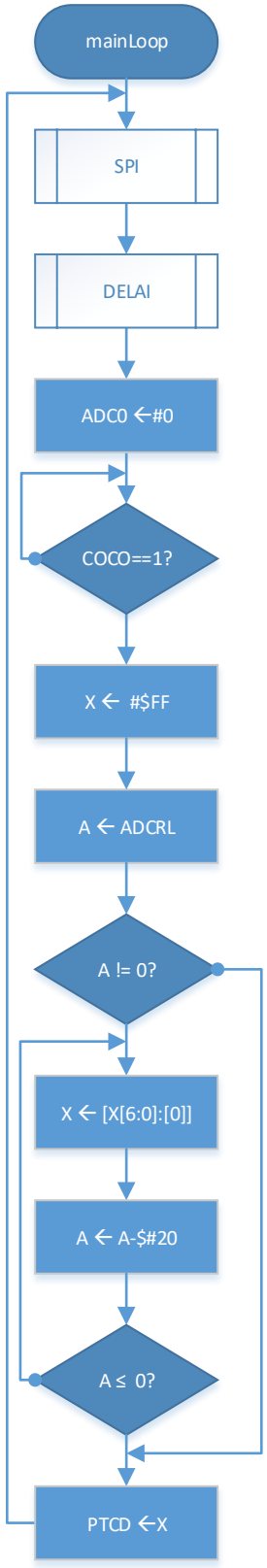
Algorigramme



Le programme ne peut fonctionner qu'après avoir réalisé plusieurs initialisations :

- Sommet de la pile
- SPI pour paramétrer le potentiomètre numérique
- CAN pour paramétrer la réception de la valeur analogique
- Espace mémoire

Cette série d'initialisations est faite dans la première partie du code source, avant le début de la boucle principale.



La mainLoop contient la boucle principale du programme.
Elle reçoit une valeur numérisée et gère l'affichage des LEDS en conséquence. L'affichage permet de visualiser la valeur analogique, envoyée par le potentiomètre, de manière linéaire.

Elle commence par appeler les sous-programmes SPI et DELAI avant d'attendre la fin d'une conversion analogique-numérique.
Ensuite le programme traduit cette valeur pour générer un affichage linéaire, compréhensible au premier coup d'œil.

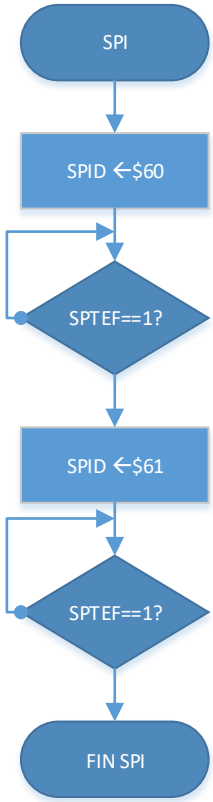
Pour effectuer cela, le programme effectue des soustractions successives sur la valeur convertie et décale à chaque fois le registre X vers la gauche.
Pour un affichage sur 8 leds, il faut faire des soustractions telles que :

$$A = A - 256/8$$

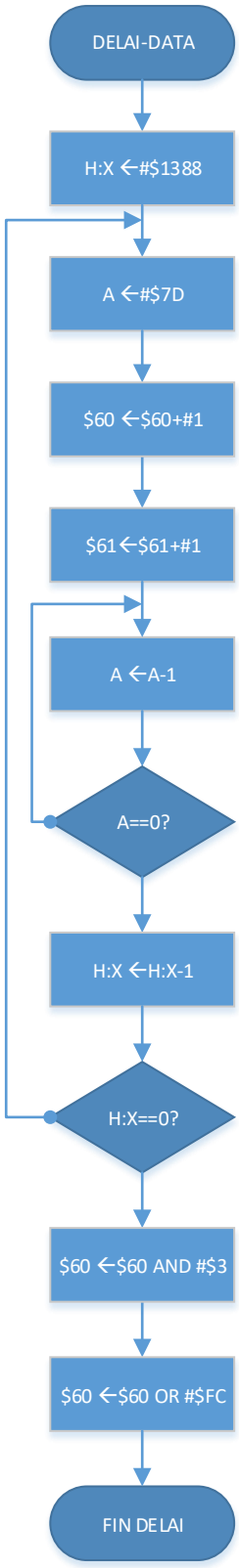
Avant la première soustraction, le programme teste si la valeur est nulle. Dans ce cas le reste de la conversion est sautée.

Enfin la valeur dans X est envoyée au port PTCD pour allumer les LEDS voulues.

Ce genre d'affichage est similaire aux jauges de batteries sur les équipements mobiles tels que les batteries externes.



Le sous-programme SPI se contente d'envoyer un double octet au potentiomètre numérique.
Le premier octet contient l'adresse mémoire à laquelle s'adresse la commande, ladite commande, et les 2 bits forts de la donnée.
Le second octet envoie les 8bits faibles de la donnée.



Ce sous-programme possède deux fonctions distinctes.

- Premièrement c'est un délai qui permet d'avoir une visualisation correcte des LEDS entre deux conversions analogiques-numérique. Sans ce délai les conversions s'enchaineraient trop vite pour l'œil humain.
- Ensuite ce délai permet de générer une donnée sur 10bits qui sera envoyée au potentiomètre par SPI.

Fonctionnement du générateur de data :
A chaque boucle externe, les mémoires hautes et basses - $60 et $61- sont incrémentées pour donner une valeur qui, sans être aléatoire, sera différente après chaque délai. A la toute fin du délai, on applique un masque à l'octet haut pour garder les deux bits de data et y réintégrer les 4 bits d'adressage et les deux bits de commande.

$$23 + 5000(4 * 127 + 20) = 2\,640\,023 = 520ms$$

Code source

```
; variable/data section
MY_ZEROPAGE: SECTION  SHORT         ; Insert here your data definition

; code section
MyCode:     SECTION
main:
_Startup:
            LDHX   #__SEG_END_SSTACK ; initialize the stack pointer
            TXS
            ; Call generated device initialization function
            JSR    MCU_init
            MOV     #$00,   $60     ;Initialise l'octet de mémoire contenant la configuration SPI du potentiomètre
            MOV     #$00,   $61     ;Initialise l'octet de mémoire contenant la donnée pour le potentiomètre num.
            MOV     #$FF,   PTCDD   ;Configure le port C, LEDS, en sortie
            MOV     #$FF,   PTCD    ;Eteint les LEDS
            MOV     #$12,   SPIC1   ;SPI : Configure la fonction SPI en maître, polarité haute, phase milieu, MSB first
            MOV     #$10,   SPIC2   ;SPI : Gestion d'esclave automatique, pas de broche unique.
            MOV     #$12,   SPIBR   ;SPI : Reglage du débit SPI : Busclock/16 : 262 144Hz
            MOV     #$00,   ADCSC2  ;CAN : Commande de conversion software
            MOV     #$60,   ADCCFG  ;CAN : Conversion sur 8bits, busclock/8

mainLoop:                       ;Programme de visualisation linéaire CAN, boucle principale
            JSR     EnvSPI          ;Appelle l'envoi SPI au potentiomètre numérique
            JSR     WAIT            ;Appelle la fonction d'attente et de modification de la DATA SPI
            MOV     #$21,   ADCSC1  ;Active une unique conversion CAN
CAN:        BRCLR   7, ADCSC1, CAN  ;Attend la réception complète d'une valeur dans le CAN pour continuer
            LDX     #$FF            ;Début de la fonction, valeur d'envoi aux LEDS tout éteint.
            LDA     ADCRL           ;Charge la valeur numerisée dans A
            CMP     #$00            ;Test de nullité de la valeur
            BEQ     FIN             ;Si la valeur est nulle, les autres tests sont sautés
LOOP:       ROLX                    ;Allumage de la LED suivante si la valeur n'est pas nulle
            SUB     #$20            ;Soustrait un huitième de la valeur max (256)
            BLS     FIN             ;le sous-programme se termine quand les soustractions successives atteignent 0.
            BRA     LOOP            ;Si 0 est atteint, le programme sort de la boucle
            STX     PTCD            ;Gestion des LEDS
FIN:        BRA     mainLoop    ;Fin, retourne au début de la boucle principale

WAIT:                           ;Sous-programme de délai pour une visualisation correcte des LEDS.
                                ;Sert aussi à générer une valeur à envoyer par SPI
            LDHX    #$1388          ;Initialise la boucle externe du délai
LOOPE:      LDA     #$7D            ;Initialise/reinitialise la boucle interne du délai
            INC     $60             ;Modifie la valeur basse de la data SPI à chaque boucle externe
            INC     $61             ;Modifie la valeur haute de la data SPI à chaque boucle externe
LOOPI:      DECA                    ;Décrémente la boucle interne
            BNE     LOOPI           ;Continue la boucle interne tant que nécéssaire
            AIX     #-1             ;Décrémente la boucle externe
            CPHX    #$0000          ;Vérifie la fin de la boucle externe
            BNE     LOOPE           ;Continue la boucle externe tant que nécéssaire
            LDA     #$03            ;Charge le masque de data de la partie haute
            AND     $60             ;Remet à 0 tout les bits non-data de la partie haute
            STA     $60             ;Replace l'octet haut en mémoire
            LDA     #$FC            ;Charge le masque de configuration de la partie haute
            ADD     $60             ;Reconfigure l'octet sans toucher aux bits de data
            STA     $61             ;Replace l'octer haut en mémoire
            RTS                 ;Retour au programme

EnvSPI:                         ;Sous-programme d'envoi des données SPI au pot num
            MOV     $60,    SPID    ;Charge l'octet haut dans le registre d'envoi
E1:         BRCLR   5, SPIS, E1     ;Attend la fin de l'envoi
            MOV     $61,    SPID    ;Charge l'octet bas dans le registre d'envoi
E2:         BRCLR   5, SPIS, E2     ;Attend la fin de l'envoi
            RTS                 ;Retour au programme
```

Conclusion

Durant la phase de fabrication, nous avons sous-estimé la difficulté accrue de la phase de soudure due au rapprochement des composants CMS. A cause de ces difficultés, des erreurs de soudure ont été commises, rendant la carte non fonctionnelle.

Dans ce projet un travail d'équipe plus important a permis de mieux affronter des difficultés nouvelles, liées aux différences entre cette dernière réalisation et les cartes précédentes.

Toutefois, malgré de nombreux tests visuels et électriques, la carte s'est avérée défectueuse au branchement